Virtual Distortion Electric Guitar Lab

Julius O. Smith III and Nelson Lee,

RealSimple Project*
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

June 5, 2008

Abstract

With familiarity with Delay Lines from previous labs under your belt, and coverage of how d'Alembert's solution to the Wave Equation can be simulated with two Delay Lines, you will now write C++ classes to simulate an electric guitar followed by an implementation of overdrive distortion. Starter code is available here.[1]

1 The String

(15 pts) Write an STK class `String.cpp` which implements a digital waveguide string model. Implement the "efficient" block diagram found here.[2] The `tick()` function should accept an input sample (for exciting the string) and return the current string output. Control parameters governing construction should include the desired pitch, the plucked position and the observing point. Other characteristics, such as string damping, can be hard-wired or brought out as controls.

The string loop filter should be a symmetric second-order FIR filter $H_l(z) = a + bz^{-1} + az^{-2}$. The string delay line should be interpolated using allpass interpolation (e.g., using the STK class `DelayA`). If hard-wired, the string loop filter coefficients should be $b = 1/2$, $a = 1/4$.

2 The Instrument

(15 pts) Writxe an STK class `ElecGuitar.cpp` which uses a single instance `String` to simulate a string vibration. Each `ElecGuitar` string should have the following features

1. It excites its String with a white-noise burst which is filtered by a one-pole lowpass filter (the default pole location should be 0.5). The noise is available from an STK class called `Noise` and its burst should have a controllable duration up to at least one period in length (the default pluck time should be 50 samples). The lowpass filter should have dc gain equal to 1

*Work supported by the Wallenberg Global Learning Network

[1] http://ccrma.stanford.edu/~jos/realsimple/electric_guitar/starter_code.tar.gz

[2] http://ccrma.stanford.edu/~jos/pasp/Equivalent_Forms.html

(or slightly less) for all "brightness" settings. It should have a `noteOn` and `noteOff` methods to deal with the start and end of a note. See examples from other instruments available in STK.

2. It should allow for pluck and magnetic pick-up positions to be specified in its corresponding `String` instance as a ratio between [0,1] of the string length from the bridge. Let the default pluck and pick-up positions be 1 and 3 inches from the bridge (respectively).

3. It should allow for a feedback input to be added to the string excitation from external programs e.g. via `setFeedback` method.

You may use as an example, `SimpString.cpp`, given in `gtrc` directory from the `jos-overlay`. Note, however, that your `String` should do nothing except setting up the string (including pluck and pick-up positions). It has to be excited through `ElecGuitar`.

3 Overdrive

(5 pts) Write an STK class `Overdrive.cpp` which simulates a nonlinear soft-clip amplifier distortion, as described in *"Amplifier Distortion + Amplifier Feedback"*.[3] The controls should be (at least) pre-distortion gain, post-distortion gain, and direct-signal gain. Its `tick` method takes an input and emits an output. In the main program, the input to the distortion unit should be the sum of all strings' outputs.

4 Amp Distortion

(5 pts) Implement the amplifier feedback, as described in *"Amplifier Distortion + Amplifier Feedback"*.[4] except that you can feedback the final output of the distortion unit for modularity. The controls are feedback gain and feedback delay. This can be simply implemented directly in the main test program `mygtr.cpp`. In the main program, the input to the amplifier feedback unit should be the output from the distortion unit. Its output is fed back into each string in `ElecGuitar`.

5 Testing the Whole Setup

5.1 Completing mygtr.cpp

(10 pts) Complete the test program `mygtr.cpp` that plays a simple SKINI score file containing a few notes using `ElecGuitar` just created. Compile and link using the `Makefile` given along with the sample SKINI file `test.ski`. Turn in your commented program listings as usual.

5.2 Playing a Melody

(20 pts) Using `test.ski`, produce examples of your electric guitar for the three different playing modes : guitar, guitar with distortion, and guitar with distortion and amplifier feedback. Concatenate these synthesized examples into a SINGLE soundfile and place a copy in your submission folder, using your last name in the soundfile name. Make sure they all sound acceptable for grading.

[3] http://ccrma.stanford.edu/~jos/SimpleStrings/Amplifier_Distortion_Amplifier.html
[4] http://ccrma.stanford.edu/~jos/SimpleStrings/Amplifier_Distortion_Amplifier.html